1

回顧

- 變數 (variables)
- 運算子
  - 算術運算子：+, -, *, /, %, ++, --
  - **關係運算子：==, !=, >, <, >=, <=**
  - **布林運算子 / 邏輯運算子：&&, ||, !**
  - 指定運算子：=

2

回顧

- 基本輸入輸出

```
int a = 3;
cout << “Hello~” << a << “\n”;
cin >> a;
```

  - 跳脫字元
  - iomanip：對輸出格式作設定
    - setprecision(n): 設定**接下來所有**輸出資料的精確度為 n
    - setw(n): 設定**下一個**輸出資料的最小寬度為 n
    - setfill(ch): 設定**接下來所有**輸出有寬度時，空白地方所填入的字元

Lecture 4

流程控制

4

今日內容

- 流程控制 (I) – if (...) else ...
  - 多重分支
  - 巢狀分支
- 流程控制 (II) – switch ... case ...

5

流程控制 – if
分支執行

Flow Control – if
Branching

6

流程控制

- 有時候，我們不太想程式裡的每一行都被執行到，而是根據現有的資料或是運算的結果，來決定程式的流程 (執行路徑)。
  - 如果成績小於 60 分 → 被當掉了，該科目要重修
  - 否則該科目過了，學分拿到。
- **if (...) { ... } else { ... }**
- **if(...) { ... } else if (...) {...} else if (...) {...} ...**
- switch (...) case ...

7

4-1.cpp

```
#include <iostream>
using namespace std;
int main() {
  int score;

  cout << "請輸入成績: ";
  cin >> score;

  if(score >= 60) {
    cout << "及格囉！" << endl;
  }
  else {
    cout << "當掉囉！" << endl;
  }
  return 0;
}
```

若 score >= 60 為真，則

否則

注意到，不是每一行程式都會被執行了 → 分支！

8

if ()

- 如果...就...不然就
  - 如果分數大於等於60分，就成績及格，不然就不及格。
  - 如果成績不及格，就要用功。
  - 所以「不然就」的部份是可以省略的。

```
if (布林值、布林運算) {
  條件成立時執行的動作
}
else {
  條件不成立時執行的動作
}
```

```
if (分數大於等於60分) {
    成績及格
}
else {
    不及格
}
```

9

4-2.cpp

```
#include <iostream>
using namespace std;

int main() {
  int a = 30;
  int b;

  cout << "\n本程式檢查 b 是不是 " << a << " 的因數。\n";
  cout << "請輸入整數 b:";
  cin >> b;
```

10

4-2.cpp

```
  cout << "\n" << b;
  if(a % b == 0) {
      cout << " 是 ";
  }
  else {
      cout << " 不是 ";
  }
  cout << a << " 的因數。" << endl;

  return 0;
}
```

11

複合關係運算

- 有時候所需的條件是複雜的, 可能需要複合多種的關係運算子 – 透過邏輯運算子來複合多個關係運算的結果
  - E.g. 如果我很餓 **而且** 我很有錢 → 吃牛排
  - E.g. 如果 我很餓 **而且** 我只有２０元 → 吃泡麵
  - E.g. 如果 我**不**餓 **或是** 我沒錢 → 不吃東西
  - 如果 (A 且 B) 則 ...
  - 如果 (A 或 B) 則 ...
  - 如果 (非 A) 則 ...

12

4-3.cpp

```
#include <iostream>
using namespace std;
int main() {
  char ans;

  cout << "你今天高興嗎 (Y/N)? ";
  cin >> ans;

  if(ans == 'Y' || ans == 'y') {
    cout << "太好了 :-) \n";
  }
  cout  << "祝你有愉快的一天!\n";

  return 0;
}
```

13

4-4.cpp

```
#include <iostream>
using namespace std;
int main() {
  int data;

  cout << "請輸入一介於 1 到 100 的數字: ";
  cin >> data;
  if(data>=1 && data <= 100) {
    cout << "謝謝您的合作 \n";
  }
  else {
    cout << "你在找我麻煩哦 \n";
  }

  return 0;
}
```

14

4-4a.cpp

```
#include <iostream>
using namespace std;
int main() {
  int data;

  cout << "請輸入一介於 1 到 100 的數字: ";
  cin >> data;
  if( data<1 || data > 100 ) {
    cout << "你在找我麻煩哦 \n";
  }
  else {
    cout << "謝謝您的合作 \n";
  }

  return 0;
}
```

15

多重條件複合時的運算順序

- if () 裡的條件敍述是從左向右分析的
  - if (A && B && C && D) {} else {}
    - 如果 A 不成立時, B, C, D 完全不會被理會
    - 如果 A 成立, B 不成立時, C, D 不會被理會
    - 如果 A, B 成立, 但 C 不成立時, D 不會被理會
    - 如果 A, B, C, 都成立時, 結果由 D 決定
  - if (A || B || C || D) {} else {}
    - 如果 A 成立時, B, C, D, 完全不會被理會
    - 如果 A 不成立, 但 B 成立時, C, D 不會被理會
    - 如果 A, B 不成立, 但 C 成立時, D 不會被理會
    - 如果 A, B, C 都不成立, 結果由 D 決定

16

多重分支

17

多重分支

- 有時，有很多不同的想要作的事情：
  - **如果**現在又渴又餓的話，就買超值全餐;
  - **否則如果**口渴的話，就買中杯可樂;
  - **否則如果**肚子餓的話，就買漢堡;
  - **否則**，就吹冷氣。

```
if (…) { … }
else if( … ) { … }
else if ( … ) { … }
else { … }
```

- 如果寫成如下，會有什麼結果？
  - **如果**現在又渴又餓的話，就買超值全餐。
  - **如果**口渴的話，就買中杯可樂。
  - **如果**肚子餓的話，就買漢堡；
  - **否則**，就吹冷氣。

18

4-5.cpp

```
#include <iostream>
using namespace std;

int main() {
  int score;

  cout << "請輸入成績: ";
  cin >> score;

  if(score >= 90) {
    cout << "A";
  }
  else if(score >= 80) {
    cout << "B";
  }
  else if(score >= 70) {
    cout << "C";
  }
  else if(score >= 60) {
    cout << "D";
  }
  else {
    cout << "F";
  }

  return 0;
}
```

19

4-5a.cpp

```
#include <iostream>
using namespace std;

int main() {
  int score;

  cout << "請輸入成績: ";
  cin >> score;

  if(score >= 90)
    cout << "A";
  else if(score >= 80)
    cout << "B";
  else if(score >= 70)
    cout << "C";
  else if(score >= 60)
    cout << "D";
  else
    cout << "F";

  return 0;
}
```

20

巢狀式判斷

Nested If

21

巢狀式(nested) 的 if 架構

```
if(條件 A) {
  if(條件 B) {
      if(條件 C) {
            (A && B && C) 時執行的動作
      }
      else {
            (A && B && (!C)) 時執行的動作
      }
  }
  else {
      (A && (!B)) 時執行的動作
  }
}
else {
  (!A) 時執行的動作
}
```

22

4-6.cpp

```
#include <iostream>
using namespace std;

int main() {
  int number;

  cout << "請輸入一個介於 1 - 10 之間的整數：";
  cin >> number;
```

23

4-6.cpp (續)

```
if(number >= 1 && number <= 10) {
   if(number%2 == 0) {
     cout << "你輸入的數字" << number << "是偶數";
   }
   else {
     cout << "你輸入的數字" << number << "是奇數";
   }
 }
 else {
     cout << "你輸入的數字" << number << "超出範圍";
 }

  return 0;
}
```

24

4-6a.cpp (改)

```
if(number >= 1 && number <= 10) {
   cout << "你輸入的數字" << number;
   if(number%2 == 0) {
     cout << "是偶數";
   }
   else {
     cout << "是奇數";
   }
}
else {
     cout << "不正確的輸入";
}

return 0;
}
```

25

巢狀 if(){} else{} 的注意事項

- 若 if, else 欲執行的動作僅一行，{} 區塊可省略，但小心出錯!!!
  - else 會與向上最靠近且未與 else 配對的 if 相關聯
- 建議都加上 {}, 以減低出錯機率

WRONG!

```
if (a > 0)
    if(b > 0)
        cout << “a>0 and b>0”;
else
    cout << “a<=0”;
```

CORRECT!

```
if (a > 0) {
    if(b > 0) {
        cout << “a>0 and b>0”;
    }
}
else {
    cout << “a<=0”;
}
```

26

if() 之用途

- 利用 if() 我們可以根據使用者輸入的資料或是計算的結果來決定那一部份的程式被執行到
- 常常我們利用 if() 來檢查使用者輸入資料的合理性 (e.g. weight >=0, age >=0, age <=200, ...))
- 利用 if() 檢查可用來根據計算的結果選擇性地執行不同的反應

27

4-7.cpp

輸入年齡可以辨認適合看什麼級別的電影

```
#include <iostream>
using namespace std;

int main() {
  int age;

  cout << "請輸入年齡: ";
  cin >> age;

  cout << "您可觀賞";
  if(age<6) {
    cout << "普通級" << endl;
  }
  else if(age<12) {
    cout << "保護級" << endl;
  }
  else if(age<18) {
    cout << "輔導級" << endl;
  }
  else {
    cout << "限制級" << endl;
  }

  return 0;
}
```

28

switch ... case

29

4-8.cpp

輸入年齡可以辨認適合看什麼級別的電影

```
#include <iostream>
using namespace std;

int main() {
  int age;

  cout << "請輸入年齡: ";
  cin >> age;

  cout << "您可觀賞";
  switch(age/6) {
    case 0:
      cout << "普通級";
      break;
    case 1:
      cout << "保護級";
      break;
    case 2:
      cout << "輔導級";
      break;
    default:
      cout << "限制級";
  }

  return 0;
}
```

30

說明

- switch(...) { case ...} 為多重分支 (branching) 的另一種寫法，可用來取代if (...) {...} else if(...) {...} else (...) {...}。
- switch(A)，其中A
  - 需為整數(short, int, long) 或是字元型態 (char) 的**變數**。
  - 決定之後的那一個分支被執行到
- case B: ... break;
  - B 為前面前A之值的可能值
  - 每一個 case 被稱為一個分支 (branch)
  - 程式在執行時會跳躍到檢查 A 變數當時的值來決定那一個分支被執行。若所有的 case 指定之 B 都與 A 之值不符合時，則程式會跳躍至 default (內定的) 分支。
  - break 很重要！switch() 只檢查一次參數，如果沒有 break 的話所有在符合條件之後的動作都會被執行到！

31

4-9.cpp

```
#include <iostream>
using namespace std;

int main() {
  char grade;

  cout << "請輸入成績 (A, B, C, D, F): ";
  cin >> grade;

  switch(grade) {
    case 'A':
      cout << "90 - 100";
      break;
```

32

4-9.cpp

```
    case 'B':
      cout << "80 - 89";
      break;
    case 'C':
      cout << "70 - 79";
      break;
    case 'D':
      cout << "60 - 69";
      break;
    default:
      cout << "0 - 59";
  }
  return 0;
}
```

33

玩玩看

- 如果把4-8.cpp 中的 break; 拿掉，程式是否可以執行？結果是否正確？
- 同樣的，如果把 4-9.cpp 中的 break; 拿掉，你是不是可以預測它的結果？

34

Fall through

- 在執行 switch (...) {case ...} 時，程式的分支判斷只執行一次，而一支分支的終點以 break; 決定之。

```
case 'B':
     cout << "80 - 89";
     break;
case 'C':
     cout << "70 - 79";
     break;
```

```
case 'B':
     cout << "80 - 89";
case 'C':
     cout << "70 - 79";
     break;
```

- 若一分支沒有 break 的話，接下來的分支亦會被執行，直到碰到 break 或是整個 switch 的區塊結束。
- switch case 中的 case 實質上只是一個標籤，而分支的動作是由 switch 執行的！！！

35

4-9a.cpp (改自 4-9.cpp)

```
switch(grade) {
  case 'A': case 'a':
    cout << "90 - 100";
    break;
  case 'B': case 'b':
    cout << "80 - 89";
    break;
  case 'C': case 'c':
    cout << "70 - 79";
    break;
```

```
  case 'D': case 'd':
    cout << "60 - 69";
    break;
  default:
    cout << "0 - 59";
}
```

36

if ... 和 switch ... 效率比較

- 在多重的 if ... else if ... 中，若判斷條件是簡單的 == 時，switch ... case ... 和 if ... else if ... 常常可以互相取代的。
- switch case 在編譯時，實際上是產生跳躍表 (jump table)，根據 switch 的值直接跳躍到對應的程式碼。
- if ... else if ... else if ... 則是在執行時期一個條件一個條件的檢查，所以執行上的效能會相對較差。

37

觀察

- 在引入 if() 和 switch() 之後, 程式的執行不再是由上而下的逐行執行了, 而可以根據程式執行當中所產生之資料而有不同的執行路徑.
- 從此, 電腦可以比一般計算機有更多的功能了!

38

作業

- 下次上課內容：迴圈 (for ..., do ..., while ...)

39

隨堂練習

1. 請使用者輸入三個數字，之後由使用者選擇讓程式計算 1) 加總、 2) 平均、3) 找最大值、或4) 找最小值，並執行相對應之功能。
   - 請使用 switch(...) { case ...} 達到不同功能選擇的目的
   - 其中找最大值與最小值時, 需在 switch case 裡面使用 if ... 判斷來找出最大值或最小值